포인터

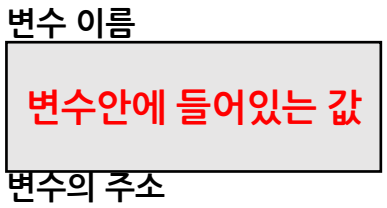
지능기전공학부 스마트기기공학전공 3학년 정건희
2023-1

1-1. 포인터에 사용되는 연산자

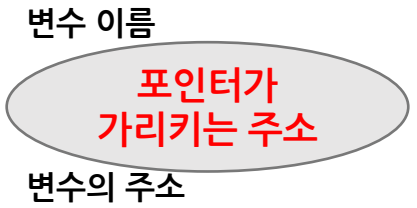
*	참조연산자 : 포인터가 가리키는 값을 접근(access)
&	주소연산자 : 해당하는 주소를 나타내주는 연산자

1-2. 포인터란?

일반 변수(int, char ...)
값을 저장하는 자료형

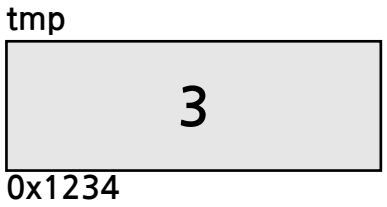


포인터 변수 (int*, char* ...)
주소를 저장하는 자료형

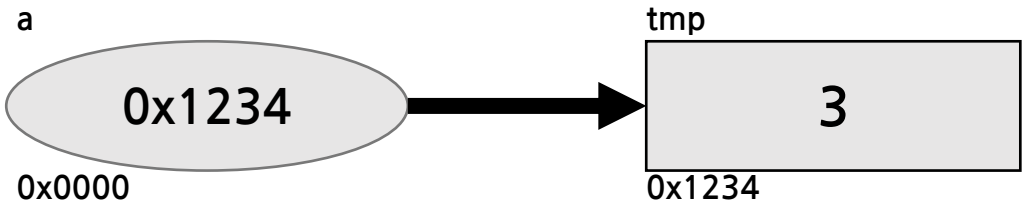


Ex)

int tmp=3;



Int *a=&tmp;



1-2. 포인터란?

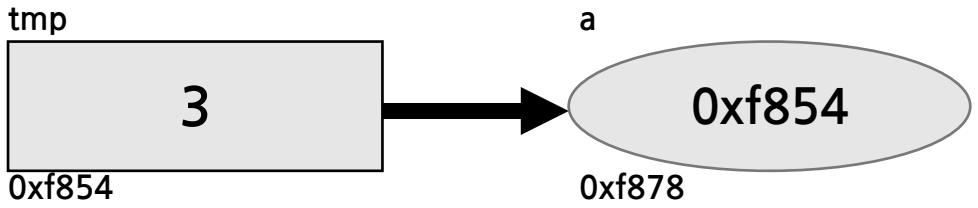
```c
#include<stdio.h>

int main() {
    int tmp = 3;
    int* a = &tmp;

    return 0;
}
```

int tmp=3;

Int *a=&tmp;



조사식 1

검색(Ctrl+E) 검색 심도: 3

이름	값	형식
tmp	3	int
&tmp	0x000000689895f854 {3}	int *
a	0x000000689895f854 {3}	int *
&a	0x000000689895f878 {0x000000689895f854 {3}}	int * *
감시할 항목 추가		

자동 로컬 조사식 1

1-3. 정리

일반 변수에는 값이 저장 되어있고
포인터 변수에는 주소가 저장 되어있다.

2-1. 포인터를 왜 사용할까? (실습1)

```c
#include<stdio.h>
void change_to_10(int a) {
    a = 10;
}
int main() {
    int tmp = 3;
    change_to_10(tmp);
    printf("%d", tmp);
    return 0;
}
```

```c
#include<stdio.h>
void change_to_10(int *a) {
    *a = 10;
}
int main() {
    int tmp = 3;
    int* adress_of_tmp = &tmp;
    change_to_10(adress_of_tmp);
    printf("%d", tmp);
    return 0;
}
```

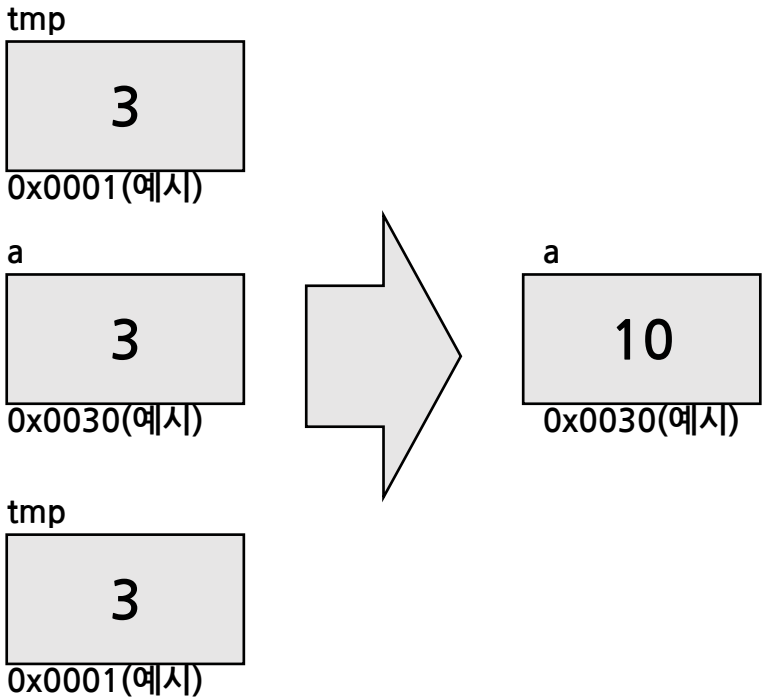
2-1. 포인터를 왜 사용할까?

```c
#include<stdio.h>
void change_to_10(int a) {
    a = 10;
}
int main() {
    int tmp = 3;
    change_to_10(tmp);
    printf("%d", tmp);
    return 0;
}
```

main 함수

change_to_10 함수

change_to_10 함수 이후



printf에서 출력되는 숫자는 3 OR 10 ??

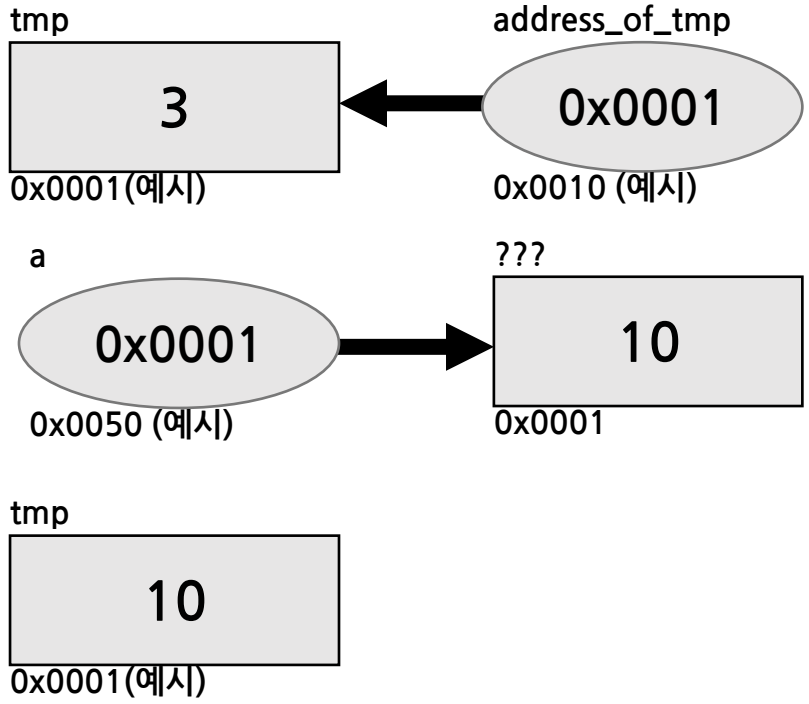
2-1. 포인터를 왜 사용할까?

```c
#include<stdio.h>
void change_to_10(int *a) {
    *a = 10;
}
int main() {
    int tmp = 3;
    int* adress_of_tmp = &tmp;
    change_to_10(adress_of_tmp);
    printf("%d", tmp);
    return 0;
}
```

main 함수

change_to_10 함수

change_to_10 함수 이후



printf에서 출력되는 숫자는 3 OR 10 ??

2-2. 실습2 (10분)

정수 1개를 입력 받은 후 square 함수를 정의하여
변수의 값을 제곱하는 프로그램을 만드시오

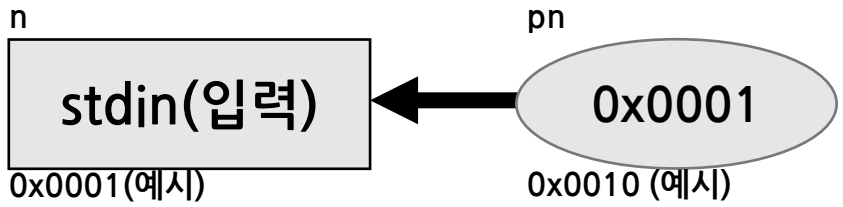
- square 함수
  - 함수 원형은 void square(int* p)
  - 입력받은 값을 제곱
  - 리턴값 : 없음

입력예시	출력예시
3	9
-3	9

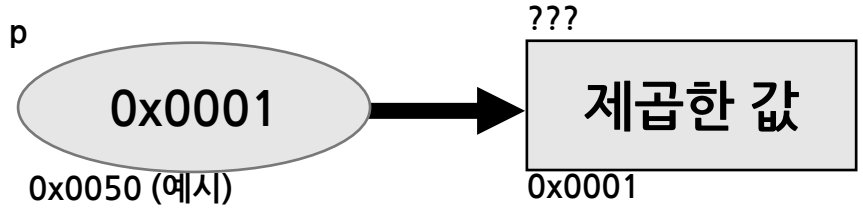
2-2. 실습2 정답

```c
#pragma warning(disable:4996)
#include<stdio.h>
void square(int *p) {
    *p = *p * *p;
}
int main() {
    int n;
    scanf("%d", &n);
    int *pn = &n;
    square(pn);
    printf("%d", n);
    return 0;
}
```

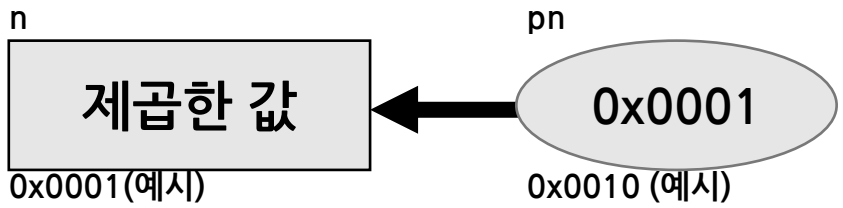
main 함수



square 함수



square 함수 이후



3-1. printf와 scanf

```
scanf("%d", a);
scanf("%d", &a);
```

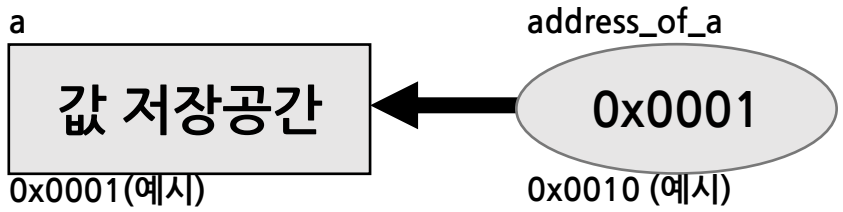
앞 실습처럼 값을 바꾸기 위해서는 주소를 이용하여 접근해야 한다!

Call by reference

3-1. printf와 scanf

```
int a;
int* address_of_a=&a;

scanf("%d", address_of_a);
scanf("%d", &a);
```



3-1. scanf 포인터로 입력받는 실습3 (10분)

1절 [문제 1][레벨 0] 다음과 같은 순서에 따라 프로그램을 작성하시오.

1) 포인터 변수 px를 이용하여 사용자로부터 값을 입력받아 x에 저장 (즉, scanf 함수의 매개변수로 포인터 변수 px 이용, 변수 x 사용 금지). 변수 y, z도 동일한 방법으로 값을 입력 받아 저장한다.
2) px, py, pz에 있는 **주소들을** py, pz, px로 이동시킨다.
3) 순서가 바뀐 최종 값을 포인터를 이용하여 출력하시오.
4) 아래 코드를 사용하시오.

```
int x, y, z;
int *px, *py, *pz, *tmp ; // tmp는 포인터이다
```

입력 예시 1	출력 예시 1
1 2 3 ↦ x	3 1 2 ↦ 한 칸씩 오른쪽으로 밀기

3-1. scanf 포인터로 입력받는 실습3 (10분)

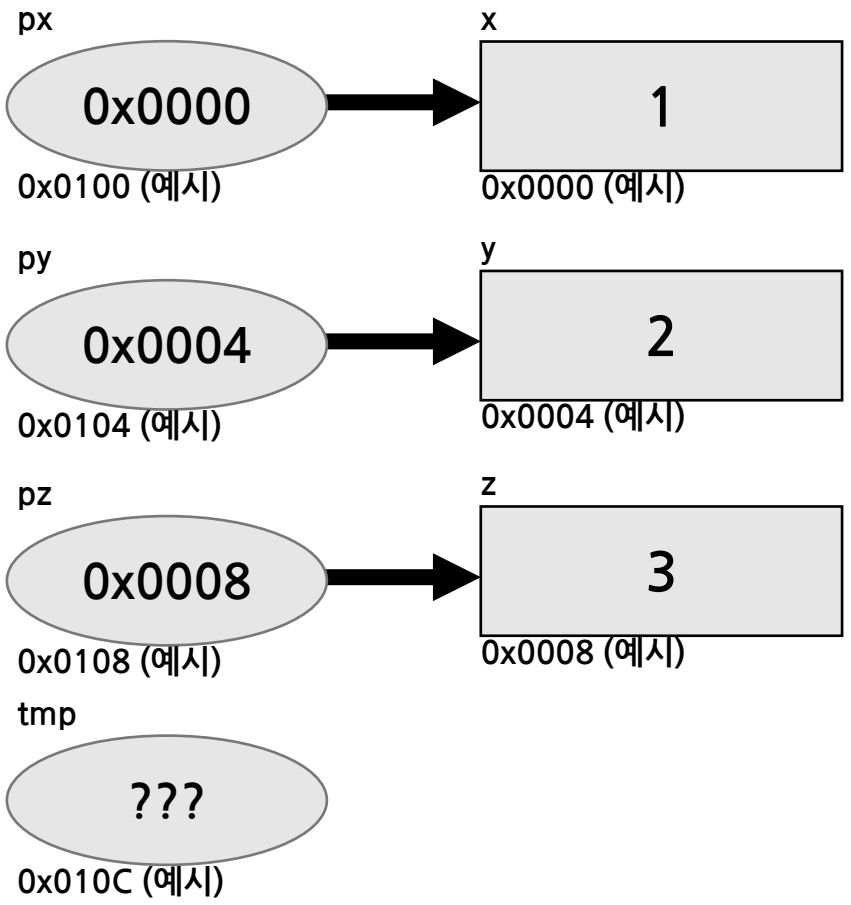
```c
int main() {

    int x, y, z;
    int* px, * py, * pz, * tmp;
    px = &x, py = &y; pz = &z;
    scanf("%d %d %d", px, py, pz);

    //swap
    tmp = pz;
    pz = py;
    py = px;
    px = tmp;

    printf("%d %d %d", *px, *py, *pz);

    return 0;
}
```



3-1. scanf 포인터로 입력받는 실습3 (10분)

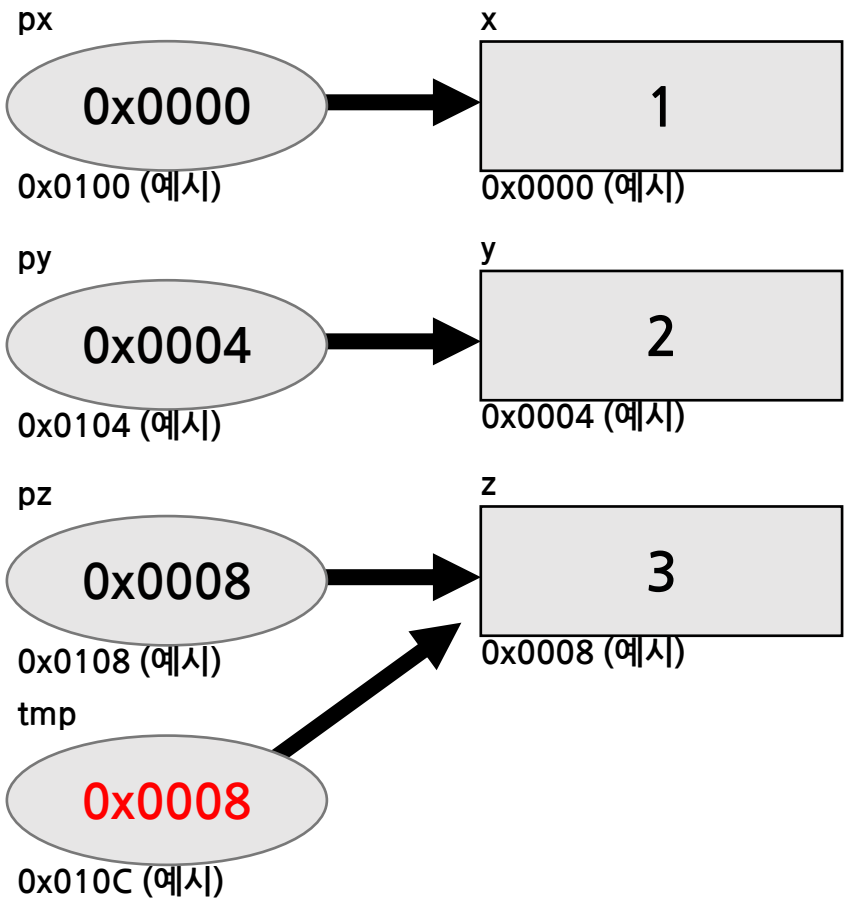
```c
int main() {

    int x, y, z;
    int* px, * py, * pz, * tmp;
    px = &x, py = &y; pz = &z;
    scanf("%d %d %d", px, py, pz);

    //swap
    tmp = pz;
    pz = py;
    py = px;
    px = tmp;

    printf("%d %d %d", *px, *py, *pz);

    return 0;
}
```



3-1. scanf 포인터로 입력받는 실습3 (10분)

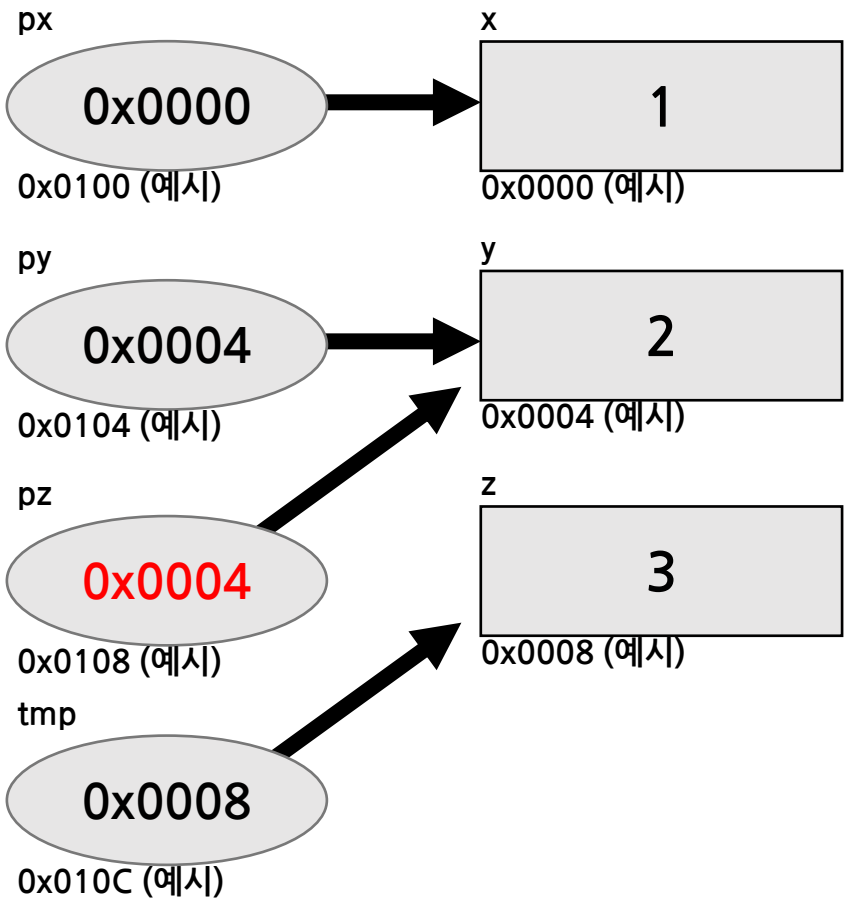
```c
int main() {

    int x, y, z;
    int* px, * py, * pz, * tmp;
    px = &x, py = &y; pz = &z;
    scanf("%d %d %d", px, py, pz);

    //swap
    tmp = pz;
    pz = py;
    py = px;
    px = tmp;

    printf("%d %d %d", *px, *py, *pz);

    return 0;
}
```



3-1. scanf 포인터로 입력받는 실습3 (10분)

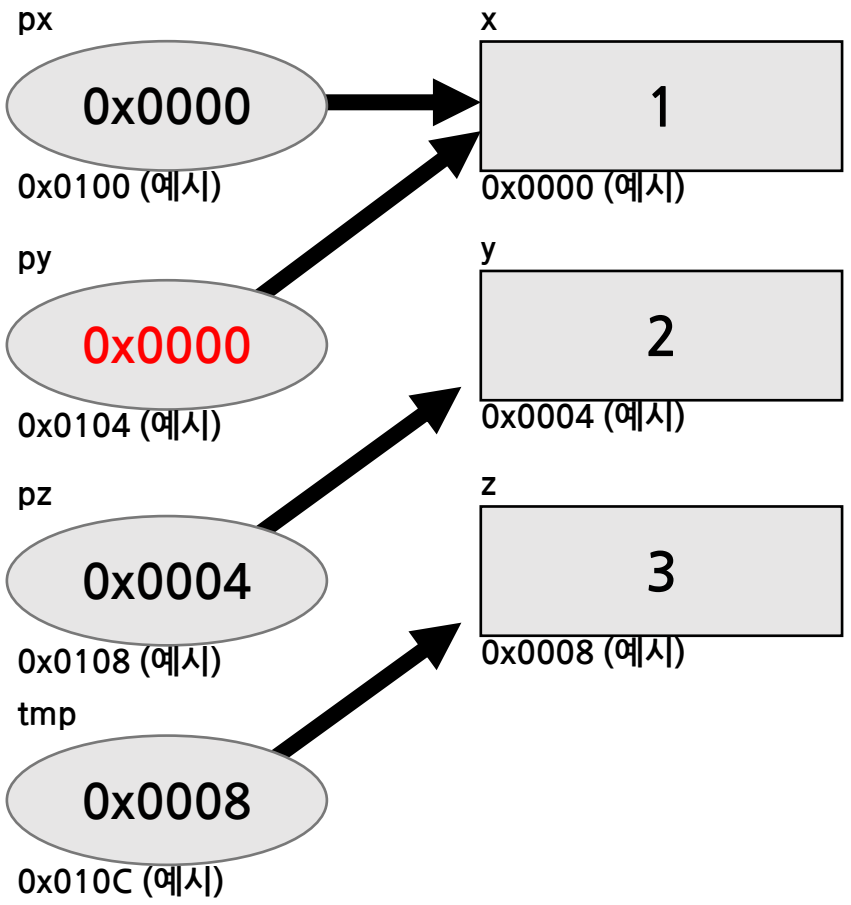
```c
int main() {

    int x, y, z;
    int* px, * py, * pz, * tmp;
    px = &x, py = &y; pz = &z;
    scanf("%d %d %d", px, py, pz);

    //swap
    tmp = pz;
    pz = py;
    py = px;
    px = tmp;

    printf("%d %d %d", *px, *py, *pz);

    return 0;
}
```



3-1. scanf 포인터로 입력받는 실습3 (10분)

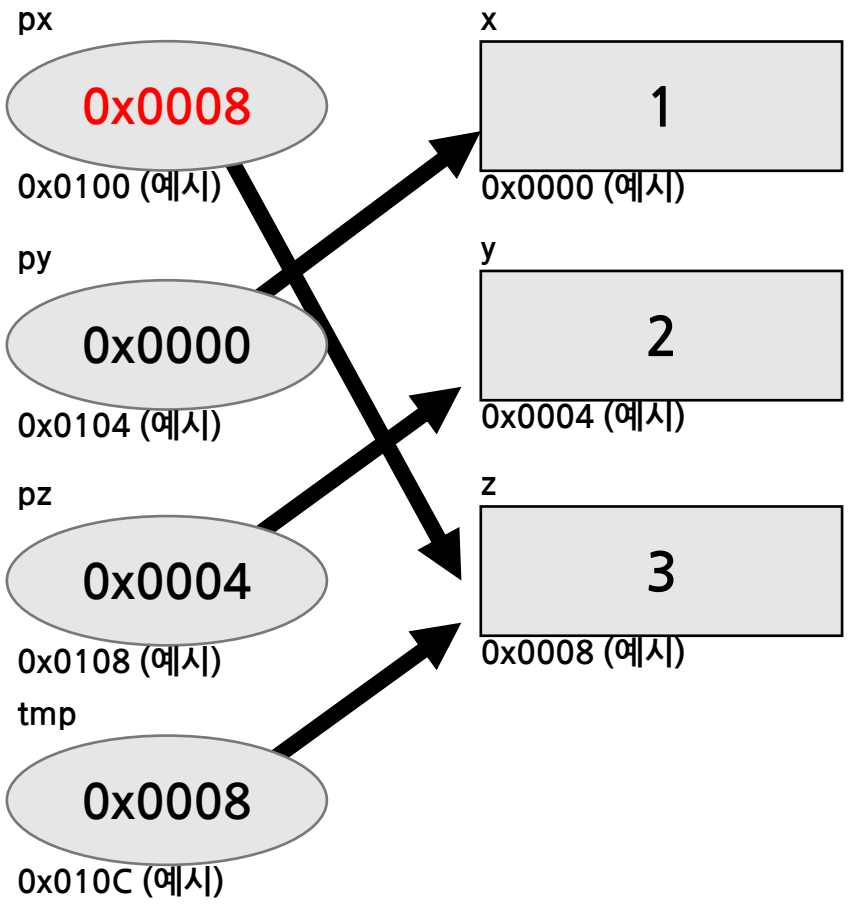
```c
int main() {

    int x, y, z;
    int* px, * py, * pz, * tmp;
    px = &x, py = &y; pz = &z;
    scanf("%d %d %d", px, py, pz);

    //swap
    tmp = pz;
    pz = py;
    py = px;
    px = tmp;

    printf("%d %d %d", *px, *py, *pz);

    return 0;
}
```



3-1. printf와 scanf

printf(“%d”, a);

변수 a의 **값**을 출력

printf(“%d”, &a);

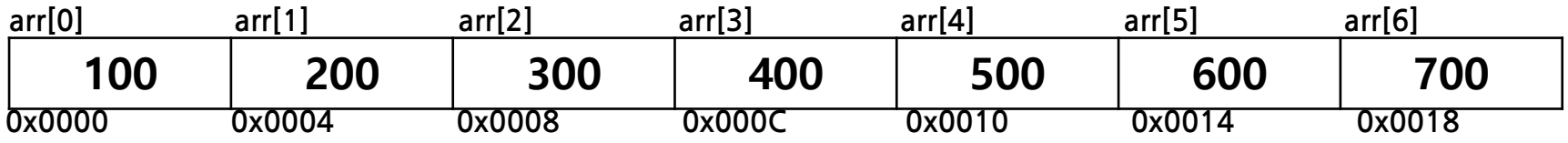
변수 a의 **주소**를 출력

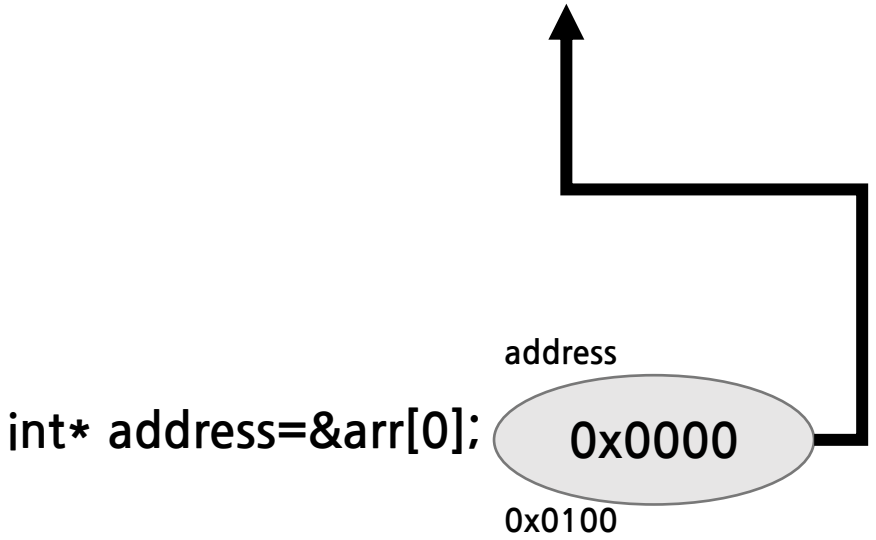
4-1. 포인터와 배열

앞장은 포인터변수와 일반변수를 연결했다면
이번 장부터는 포인터와 배열을 연결

4-1. 포인터와 배열

int arr[7];





위의 도식처럼 포인터변수에 배열의 주소를 연결한 경우 포인터를 배열처럼 사용가능

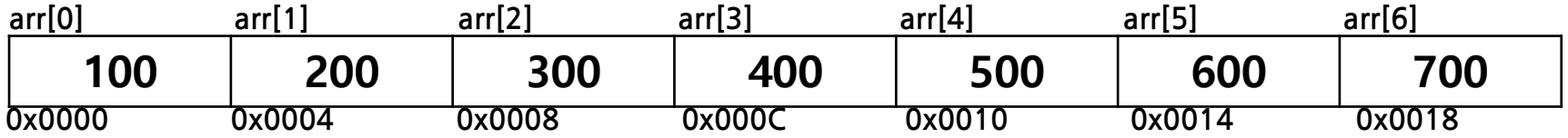
printf("%d", address[5]);

address[5]=10; // 수정도 가능

수정이 왜 될까? address[5]=*(address+5)

4-2. for문

int arr[7];



배열의 index를 이용하여 순환

```
for (int i = 0; i < 7; i++) {
    printf("%d", arr[i]);
}
```

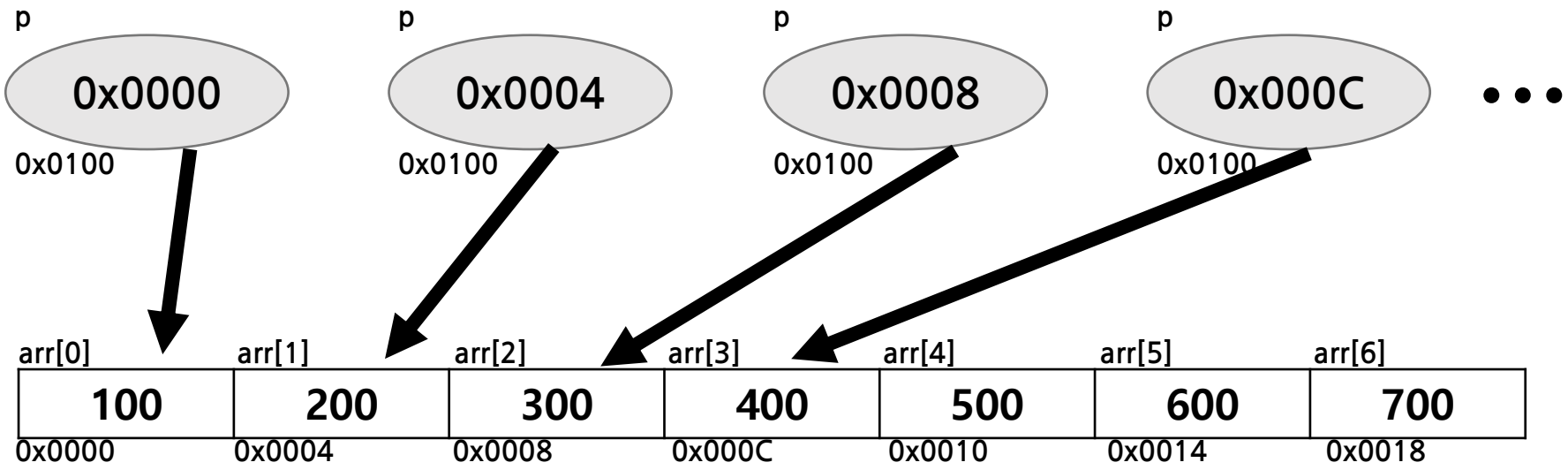
그럼 배열이 가지고 있는 주소로도 순환을 할 수 있지 않을까?

4-2. for문

배열의 주소를 이용하여 순환 (중요!!)

```
for (int* p = &arr[0]; p <= arr + 7; p++)
    printf("%d\n", *p);
}
```

int* p=&arr[0];



int arr[7];

5. QnA

Q&A